Future Timelines: Extraction and Visualization of Future-Related Content From News Articles

Juwal Regev

January 8, 2024

Abstract

In today's rapidly evolving world, maintaining a comprehensive overview of the future landscape is essential for staying competitive and making informed decisions. However, given the large volume of daily news, manually obtaining a thorough overview of an entity's future prospects is quite challenging. To address this, we present a system designed to automatically extract and summarize future-related information of a queried entity from news articles. Our approach utilizes a novel and publicly accessible multi-source dataset comprising 6,800 annotated sentences to fine-tune a language model to identify future-related sentences. We then use topic modeling to extract the main topics from the data and rank them by relevance as well as present them on an interactive timeline. User evaluations have shown that the timelines and summaries our system produces are useful. The system is available as a web application at: `https://chronicle2050.regevson.com`.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A vast amount of news articles is published on the web every single day. A significant amount of those articles contains information that is predictive or related to future events. Extracting and analyzing such information regarding a specific entity would offer a comprehensive overview of its future prospects. However doing this manually is not feasible as one would have to sift through the articles line by line, identifying nuanced references to the future.
Furthermore, this would have to be done regularly so as to always have the most up-to-date information. Thus, it is clear that an automated system, capable of extracting, processing and visualizing this information is needed. This system would be beneficial in a variety of different fields. From analyzing predictions related to stock market movements, monitoring corporate developments as well as gaining insights into market trends and emerging technologies, the potential applications are limitless.

To address this problem, a system was developed for extracting, summarizing, ranking and visualizing future-related content within news collections. It consists of five main components: 1) Data Retrieval and Preprocessing, 2) Classification, 3) Topic Modeling, 4) Postprocessing, 5) Time-Tagging. A visualization of the workflow can be seen in Figure 7.1.
The process is initiated by a user providing an entity, whose future should be explored. The system then downloads and preprocesses thousands of news articles related to the provided entity. In the next step, a neural network classifier is employed to classify sentences as either being future-related or not. Following classification, these sentences are organized into distinct topics, which are then presented to the user in a segmented manner. Additionally, the topics are labeled with representative keywords that provide insight into the content of each cluster. To provide an even better overview, we incorporate a temporal perspective into the presentation of the results. We do this by analyzing the sentences for temporal expressions, extracting and normalizing them to a date and then mapping the sentences, ranked by their relevance, onto a timeline.

In the following chapters, we explore the system's components in detail, including their implementation details and the rationale behind the design decisions made. Our goal is to provide a comprehensive overview of the system, including the inner workings and theoretical foundations of the models employed.
In Chapter 2 we begin by conducting a literature review of existing work on future information extraction and timeline generation, including both traditional and state-of-the-art approaches.
Chapter 3 delves into the task of data collection and preprocessing. It describes the different sources used to create a novel dataset of future-related and non-future related sentences and the process used to label the sentences in the dataset.
In Chapter 4, we provide a detailed explanation of the inner workings of the Transformer architecture, alongside a breakdown of both the BERT and RoBERTa models, before finally arriving at the DistilROBERTa model, which is the model used in our classifier. This section guides the reader from the foundational Transformer architecture to the final classification model.
Chapter 5 focuses on topic modeling, a technique that enables clustering of sentences into distinct topics. We briefly analyze conventional modeling approaches, discuss advanced alternatives like BERTopic, and showcase how BERTopic was configured for our system.
The extraction and normalization of temporal expressions is the topic of Chapter 6 where we explore the SUTime time tagger's capabilities and limitations.
Chapter 7 offers an analysis of the system's implementation. We delve into implementation details of the frontend as well as the backend and explain the rationale behind the frameworks that were selected.
Finally, Chapter 8 summarizes the evaluation of the system based on a user study. This assessment provides insights into the system's effectiveness and areas for further enhancement.

Chapter 2

Literature Review

2.1 Future Information Retrieval

The field of information retrieval has made substantial progress in the last few decades, particularly in the context of future-oriented information retrieval and prediction. A notable beginning was marked by the visionary paper *Searching the future* (2005) [1], which pioneered the concept of extracting temporal information from news sources and combining it with standard full-text retrieval to answer queries that integrate text and time. The paper's main contribution is defining the future retrieval problem and demonstrating its practicality through initial experiments and probabilistic models.

The methods for future information retrieval that followed, relied on time-taggers and predefined temporal expressions to extract data. For example the year 2000 brought a temporal tagger [2] that identified temporal expressions from text, tailored to news. It used hand-crafted rules as well as some machine learning to identify time expressions and assign normalized values.
ChronoSeeker [3] also utilized a rule-based approach to identify future-related sentences by employing chrono words (specific patterns and keywords typical of future-oriented sentences) combined with future years (e.g., "by the year 2040," "2010-2050") to formulate search queries. Sentences containing these chrono words were filtered from web pages and were used to extract features for training a machine learning model that accurately detected future-related sentences. A similar approach was followed in *Analyzing Collective View of Future, Time-Referenced Events on the Web* (2010) [4], where future-related sentences were identified using regular expressions like: temp_modifier+(the)year(s)+yyyy, which would match sentences like "In the year yyyy ..." or "... by the year yyyy." *Ranking Related News Predictions* (2011) [5] also used hand crafted rules, this time with the TARSQI toolkit which extracts temporal expressions from documents.

Several studies have extended the traditional approach by analyzing obtained future-related sentences for distinctive characteristics and then doing classification based on those.
Improving Retrieval of Future-Related Information in Text Collections (2011) [6] for example took an approach focusing on retrieving time-referenced and time-unreferenced predictions from text collections. Particularly the latter one is challenging as one has to move beyond rule-based methods for detection. The approach involves assembling document collections that represent past and future references. Then characteristic terms are identified that are prevalent in these collections. When analyzing a sentence, its classification as future-related or not is determined based on the presence of these characteristic terms.
Computational Exploration of the Linguistic Structures of Future: Classification and Categorization (2015) [7] used a combination of constituency grammar parsing, n-gram features, and 39 predefined syntactic rules to form features used to train an ADAGRAD classifier for detecting future-oriented sentences. *Automatic Extraction of References to Future Events from News Articles Using Semantic and Morphological Information* (2015) [8] used morphosemantic patterns as features, while *Extracting Predictive Statements with Their Scope from News Articles* (2018) [9] classified clauses using features like POS tags and word co-occurrences to determine if a sentence refers to the future. Semantic role labeling was also employed for this purpose in *A Method for Extraction of Future Reference Sentences Based on Semantic Role Labeling* (2016) [10].

Some systems employed a predictive approach that analyzed past data to forecast future events.
Predicting the News of Tomorrow Using Patterns in Web Search Queries (2008) [11] introduced PROFET, a method leveraging large-scale web resources like Google Trends to predict 100 terms that would dominate news headlines up to a week ahead. It underscored the potential of utilizing user search behavior as a predictive tool for future events. The paper *Mining the Web to Predict Future Events* (2013) [12] also analyzes historical news data to forecast significant occurrences in the future. News stories spanning over two decades were analyzed to

extract event sequences. These sequences were then clustered together based on their shared topics using topic tracking algorithms. This resulted in the construction of event storylines. The storylines were then enriched with factual data from the web.

2.2 Timeline Summarization

In the timeline summarization line of research, which is also related to our study, content is grouped and ranked to form a timeline. For this there have also been multiple different approaches over the years.

Automatic Generation of Overview Timelines (2000) [13] introduced a model for automatically creating timelines from text corpora. The process starts with extracting named entities and noun phrases from the timestamped articles. A chi-square test assesses the significance of term occurrences. Peaks are identified by finding runs of days with high chi-square scores. Terms with overlapping peak date ranges are clustered, representing related topics or events. Each cluster is labeled using the most prominent named entity and noun phrase. The clustering evaluation reveals disagreements on what constitutes a "topic".
The method proposed in *Extracting Collective Expectations about the Future from Large Text Collections* (2011) [14] employs a time-tagger to identify and normalize temporal expressions in news articles. A probability distribution is assigned to each temporal expression to represent the uncertainty associated with the future event implied by the detected temporal expression. Subsequently, each sentence along with its corresponding temporal expression is represented as a bag of words and the assigned probability distribution. These representations are then clustered using a Gaussian Mixture Model to group similar future events based on their textual content and similar probability distributions.
Temporal Summaries of New Topics (2001) [15] presented an approach to generate summaries that capture key events within a news topic over time, using probabilistic models for "novelty" and "usefulness". It showed that simple probabilistic models performed reasonably well, but not dramatically better than baseline models.
Query Based Event Extraction along a Timeline (2004) [16] further refined this concept by extracting sentences relevant to a query and placing them along a timeline based on their publication date. At each date the top N sentences are selected using special probabilistic measures like "interest" and "burstiness".
In the paper *Timeline Generation through Evolutionary Trans-Temporal Summarization* (2011) [17] a new technique for Evolutionary Trans-Temporal Summarization was presented. It addresses the challenge of tracking the progression of news stories over time. Key steps include gathering time-stamped web documents, analyzing sentence dependencies within and across dates by using models for affinity and diversity among sentences to calculate their global and local ranking scores using DivRank (a variation of the PageRank algorithm). These scores help in constructing a ranking for sentences that are most important across the entire timeline and within individual dates. Based on the ranking, the most relevant sentences are selected to represent key points in the news evolution.

The first abstractive approach to timeline summarization came with *Abstractive Timeline Summarization* (2019) [18]. Instead of identifying important sentences from a corpus and copying them directly into a timeline, this approach generates new sentences that combine information from various sources. It does this by first clustering sentences into topics using Affinity Propagation. The most frequently occurring date in each cluster is identified as the cluster's representative date. For each cluster, candidate summary sentences are generated using Multi-Sentence Compression, which fuses information from multiple sentences into a single sentence. The candidate sentences are then ranked based on date relevance, informativeness and linguistic quality. The top-ranked sentences are selected greedily to form the timeline summary.

More recently Multi-Timeline Summarization developments improve upon standard timeline summarization methods by generating multiple, high-quality and diverse timelines that summarize different stories. *Multi-TimeLine Summarization (MTLS): Improving Timeline Summarization by Generating Multiple Summaries* (2021) [19] proposed an unsupervised method that clusters SBERT embeddings through Affinity Propagation to detect events. This is followed by the selection of important events. Events that frequently co-occur and reference each other are identified in an event linking step and will form their own timeline. For each timeline, only the exemplar sentence is taken from each constituent event and the top 3 frequent non-stop words in each timeline are extracted to characterize the timeline topic.

Chapter 3

Dataset

For our system to work, a classifier must be developed that is capable of accurately classifying sentences as either future-related or non-future-related. The classifier should be implemented in the form of a neural network. For these types of models to perform well, they have to be trained on a lot of data. Another thing to consider is that a neural network is only as good as its training data because it can only learn from the data it is given. That is why it's crucial that this data is diverse. When a network is trained on diverse data, it is exposed to different patterns within the data, which makes it understand the underlying structure and relationships better. [20] To make this concrete, it means that we should provide our network with sentences about different topics, with different words and different structures. That is why our approach to data gathering differs from traditional methods. As already seen, earlier strategies have frequently relied on extracting data using temporal expressions. However, this approach has its limitations. Predictions can often be intricate, lacking explicit dates or simple temporal cues. To address this, our dataset (which is publicly available[1]) incorporates a rich variety of 6,800 manually labeled sentences. They have been collected from different sources without relying on specific queries, and therefore they exhibit unique lexical and structural features and contain a diverse set of topics. This enables our classifier to identify future-related content that does not contain standard temporal expressions hinting at future references.

3.1 Sources of Dataset

Our data originates from four different sources. A detailed breakdown of them is provided in Table 3.1. In the following, the four sources are described in more detail.

Sources	**Positives**	**Negatives**	**Sentences without temporal expressions (%)**
Longbets	448	0	20
Horizons	51	62	84
ChatGPT	305	305	70
News	2501	3128	39
Total	**3305**	**3495**	**41**

Table 3.1: Data Source Analysis: Count of Future-Related (Positive) vs. Non-Future-Related (Negative) Sentences, along with Percentage Free from Temporal Expressions.

3.1.1 LONGBETS

First we looked for websites where users can make predictions about the future in the hopes of getting sentences from various different topics that give information about future events. Longbets[2] is a platform where users can make predictions about various topics of interest. The predictions on Longbets, spanning from realistic forecasts about politics and technology to more bizarre ones, extend all the way from the near future into the next century. On the website each prediction has to be summarized in one sentence. Additionally there is an option to give more context in the description section and even a possiblity to bet real money on the prediction. In the following we see several examples of sentences that have been scraped. Figure 3.1 shows a screenshot of the website.

[1] `https://github.com/regevson/chronicle2050`
[2] `https://longbets.org`

NO.	PREDICTION	DURATION	STAKES	PREDICTOR	CHALLENGER
933.	By 2043 there will be archaeological discoveries demonstrating equal or greater architectural advancement to the monuments at Göbekli Tepe, dating from before 11,000 BC.	20 years 02023-02043	$400 $200	Samo Burja	Scott Alexander
930.	By 2035 there will be at least one public protest of more than 100 people in a G7 county against the switching off of a machine or software program on the grounds that it is sentient and has a right to live.	12 years 02023-02035	$400 $200	William E Hanbury	Ansgar J Brenninkmeije
887.	By January 1st, 2037, Tesla will have been the first company with 1 million vehicles that are capable of SAE Level 4 autonomy on over 90% of public roads in the contiguous United States, with human-level safety or better, and this capability will be usable by the general public commercially.	15 years 02022-02037	$400 $200	Yarrow Bouchard	Michael P Dierickx
883.	An annual average temperature anomaly value above the 1850-1899 baseline will be published in the Berkeley Earth Global Temperature series as 2.0C or higher on or before the 2037 value (published in 2038).	17 years 02021-02038	$1,000 $500	John Mitchell	Zeke Hausfather
868.	Private Nonfarm business productivity growth will average over 1.8 percent per year from the first quarter (Q1) of 2020 to the last quarter of 2029 (Q4).	9 years 02021-02030	$400 $200	Erik Brynjolfsson	Robert J Gordon

Figure 3.1: Screenshot of the LONGBETS website.

- *"By 2030, commercial passengers will routinely fly in pilotless planes."*
- *"Over the next ten years, we will make measurable global progress in all five areas of the human condition: food, access to clean water, health, education, and the price of energy."*
- *"By 2029 no computer - or 'machine intelligence' - will have passed the Turing Test."*

Analysis of the dataset

Around one thousand predictions were obtained from Longbets. The majority of them is one sentence long.

- 87% of them contain a date
- 98% of them contain *"will"*

As we can see, this dataset does not meet the optimal criteria we established earlier, as it lacks the desirable attributes of diverse sentence structure and words. The majority of sentences have a structure like *[... will ... -some date-]*. The content of the sentences does however represent different areas and topics. To create a balanced dataset, one thousand negative sentences scraped from news articles were added. In order to maintain structural consistency with the positive sentences (which mostly contained dates), only negative sentences containing dates were selected.

3.1.2 Horizons

Horizons[3] is more focused on emerging technologies. It is an "interactive visualization tool for discovering and learning about the technologies affecting our future". It introduces 100 emerging technologies by first describing them in detail and then focusing on their "Future Perspectives". For every technology the content was scraped and stored. It was subsequently manually reviewed for future-related content. To make the dataset balanced, non-future-related sentences were added to the dataset. These sentences were taken from the technology description section of the same website. Figure 3.2 depicts a screenshot of the Horizons website.

[3] `https://radar.envisioning.io/horizons`

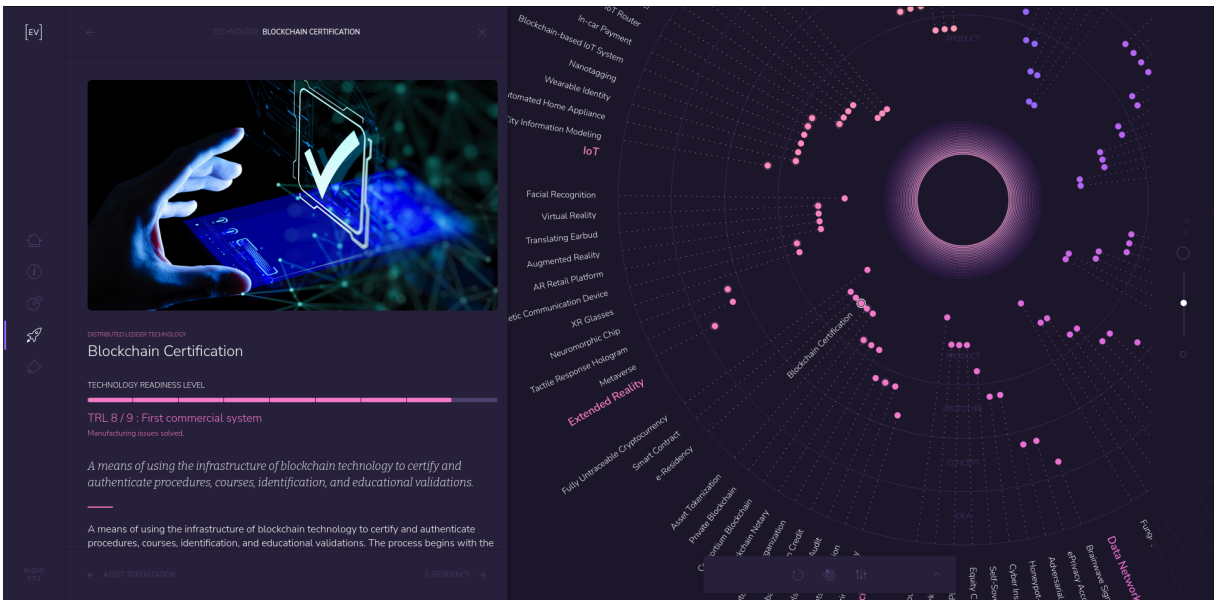

Figure 3.2: Screenshot of the Horizons website.

Analysis of the dataset

Approximately 50 sentences were obtained as a result of this process. This is not a lot, however the quality of those sentences is high. Unlike the sentences scraped from Longbets, these sentences have a more diverse structure with predictions being formed using words such as *could*, *may*, *should*. The only drawback is that all sentences are about technology. Here are the statistics of the Horizons dataset:

- 23% of them contain a date
- 47% of them contain *"will"*

3.1.3 ChatGPT

ChatGPT[4] is a groundbreaking language model developed by OpenAI in 2022. As a large-scale language model, it has been engineered to provide human-like responses to prompts and questions, mimicking a natural chat-based conversation.
The architecture of ChatGPT is based on the GPT-3.5 model, but newer iterations like GPT-4[5] have since been introduced. Its underlying framework, the transformer architecture, plays a crucial role in enabling ChatGPT to understand and generate coherent and contextually appropriate responses.
By training on vast amounts of text data, ChatGPT has gained an impressive understanding of various domains, allowing it to engage in discussions on diverse topics. From addressing general knowledge questions to offering opinions and insights, ChatGPT demonstrates a remarkable ability to communicate effectively. [21]

ChatGPT for Training Data Generation

ChatGPTs ability to generate responses in a chat-like format makes it an excellent tool for creating high-quality training data. By providing specific prompts to ChatGPT, one can obtain responses that closely align with the desired data requirements. This level of control allows for the generation of highly specific and tailored training data.

Generating Future-related Sentences with ChatGPT

As previously mentioned, the sentences extracted from sources such as *Longbets* exhibit a high similarity in structure, typically characterized by the presence of the word *"will"* followed by a future date. Due to the necessity for a more diverse range of sentence structures and vocabulary, the integration of ChatGPT became necessary. By leveraging the chatbots capabilities, sentences containing diverse sets of words and having diverse sentence structure were generated. The incorporation of ChatGPT fulfilled the requirement for diversity in the

[4] `https://chat.openai.com/`
[5] `https://openai.com/research/gpt-4`

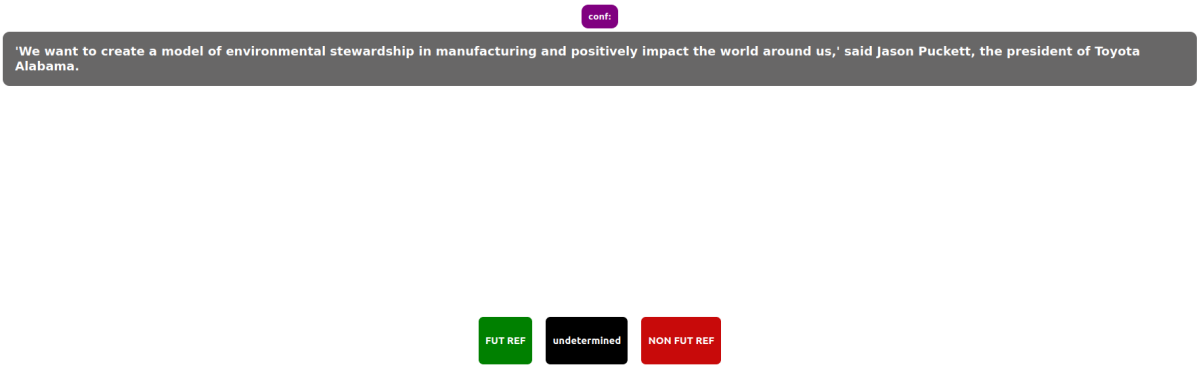

Figure 3.3: Screenshot of the classification website.

training data.

In the following a subset of prompts are shown, that were given to ChatGPT:

- *"Generate X sentences about various topics, that contain future related information. Try to mix their sentence structure up and use different words."*
- *"They should use words like 'could, 'would', 'may', 'might', ..."*
- *"They should be about two lines long."*

3.1.4 News

As already layed out, the classifier will ultimately need to classify sentences extracted from news articles. Therefore it makes sense for the dataset to be primarily composed of sentences originating from a news corpus. Manually searching for and extracting future-related sentences from news articles can be very time-consuming however. Consequently a bootstrapping approach was followed, which involved training the classifier initially only on the Longbets, Horizons, and ChatGPT datasets. This first version of the classifier was then used to classify sentences extracted from 20,000 New York Times[6] articles. Despite the limited amount of data used in the training process, the resulting classifier demonstrated an ability to make accurate predictions, making it much more viable to extract and review further predictions than extracting them manually. Moreover this gave the chance to identify the types of sentences that the model was uncertain about, which were then manually classified and incorporated into the training process.

3.2 Annotation of Future-related Sentences

During the annotation process, manual annotation was required only for the news articles and horizons sources. All other sources, including Longbets and ChatGPT, consisted entirely of content related to the future.
To streamline the annotation process, a simple website was developed specifically for this classification purpose. The website interface, depicted in Figure 3.3, is designed to enable efficient labeling. At the top of the page, the sentence to be classified is displayed. Additionally, the confidence level of the classifier regarding the sentence's future-related content is presented below the sentence. This feature aims to provide insights into the classifier's performance regarding different sentence types.
Below the confidence level, three buttons are displayed. The left button allows the user to label the sentence as containing future-related information. The middle button served as an option when the user is unsure about the appropriate category for the sentence. The right button enables the user to label the sentence as not containing future-related information.
This website-based approach significantly simplified the annotation process, offering a user-friendly environment for efficient classification.

[6] https://www.nytimes.com/

Chapter 4

Classifier

In this chapter, we delve into the core component of this project - the finetuned DistilRoBERTa classifier. However, before delving into DistilRoBERTa itself, it is essential to first understand the foundations upon which it is built. Therefore, we begin by examining the Transformer, BERT and RoBERTa models, as DistrilRoBERTa builds upon their advancements. Then we explore the motivations behind selecting DistilRoBERTa for our classifier and proceed to unravel the intricacies of its architecture. This includes an in-depth exploration of its inner workings, the specifics of its training data, and the methodology employed during its training. Most importantly, we investigate in detail the process of tailoring the DistilRoBERTa model to suit our unique classification task. Finally, we conclude our discussion by analyzing the evaluation and performance of the finetuned classifier.

4.1 Transformer Architecture

The Transformer model, introduced in the influential paper *Attention is All You Need* (2017) [22], represents a significant breakthrough in deep learning models. It emerged as a solution to tackle sequence-to-sequence tasks, with machine translation as its primary application. Prior to the Transformer's introduction, recurrent neural networks (RNNs) such as LSTMs dominated the field of sequence modeling and transduction. However, these RNN-based models posed challenges due to their sequential nature, which hindered efficient parallelization. As sequence length increases, the sequential computations of RNNs become a bottleneck. This limitation prompted the need for a more parallelizable approach. The Transformer architecture effectively addresses this parallelization problem by leveraging a novel technique known as self-attention. Self-attention enables the Transformer model to process sequences in a highly parallelized manner. Unlike RNNs, which depend on the sequential processing of each time step, self-attention allows for simultaneous computation across the entire sequence. This breakthrough concept revolutionized the field by efficiently capturing dependencies between different positions in the sequence. [22] Today the Transformer has found utility across a wide range of domains. It is extensively employed in various areas of deep learning and natural language processing. [23]
In the following sections, we delve into the basic architecture of the Transformer model and explore the self-attention mechanism.

4.1.1 Self-Attention

As seen in Figure 4.1, the Transformer consists of an encoder (left block) and a decoder (right block). As the BERT model only uses the encoder-part, we will only look at the inner workings of the encoder. We will start with the Self-Attention mechanism. Self-attention is a fundamental component of the Transformer model. It allows the model to capture relationships between different positions in a sequence. A concrete example of such a sequence is a sentence. Before computing the self-attention, the words within the sentence are transformed into vectors through an embedding algorithm. Consequently, each word is represented by a vector, forming a sequence of vectors. After that, positional encoding is applied to each input embedding vector by adding a positional encoding vector. These vectors represent specific positions in the input sequence, allowing the model to understand the order of the input data and capture sequential relationships. By incorporating positional encoding, the model can effectively process and interpret the words based on their position in the sentence.
Within the initial encoder, the sequence of vectors undergoes processing through two consecutive layers: the self-attention layer and the feed-forward layer. In the self-attention layer, each vector is split into three distinct vectors: a query-vector (Q), a key-vector (K), and a value-vector (V). These vectors are obtained by multiplying the input vector with three learned matrices, with each matrix dedicated to generating one of the respective vectors. These matrices are trained by the model during the learning process.

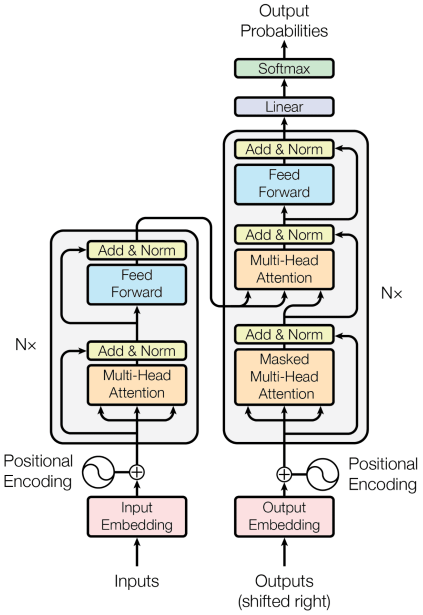

Figure 4.1: Architecture of the Transformer model. [22]

As a result, for each word in the sequence, we obtain three corresponding vectors: Q, K, and V. To compute the self-attention for a specific word, the dot product between its query-vector (Q) and the key-vectors (K) of all other words is calculated. This dot product operation produces similarity scores, representing the degree of similarity or relevance between the word and other words in the sequence.
To stabilize the gradients and ensure effective learning, the similarity scores are scaled. Additionally, a softmax function is applied to these scaled scores to normalize them. The softmax operation produces weights that indicate the relative influence or importance of each word on the word for which the self-attention is being computed.
Each value-vector (V) is then multiplied by its corresponding softmax-weight, and the resulting values are summed up. This process generates a new vector for the word under consideration. The new vector (computed through self-attention) encapsulates the original word's representation while incorporating the contextual information from the other words in the sentence. This integration of context within the new vector therefore captures the interdependencies and relationships among the words. The outlined process is done for all words in the sentence. [22]

4.1.2 Multi-Head Self-Attention

Instead of generating a single attention vector for each input embedding, the original Transformer model utilizes a technique known as multi-head self-attention, where multiple attention vectors are computed simultaneously. Typically, eight attention vectors are calculated in parallel.
To accomplish this, the model employs multiple sets of Q, V, and K matrices, with the exact number determined by the number of attention heads used. Each input word embedding undergoes the self-attention process multiple times, with each iteration utilizing a distinct set of Q, V, and K matrices.
As a result, each word embedding in the sequence is associated with multiple attention vectors. To integrate the information from these attention vectors, they are concatenated together. The concatenated vector is then passed through the subsequent neural network layer, which follows the multi-head self-attention layer. One reason for the use of multiple attention heads is that the multiple attention heads allow the model to focus on different aspects of the input sequence. Thus the model can capture various patterns, relationships and dependencies among the data. This allows for a more comprehensive representation of the input, enabling the model to learn more nuanced and detailed information. [22]

4.2 BERT

Having gained familiarity with the encoder part of the Transformer model, we now turn our attention to a powerful application of this architecture. In this section, we delve into the inner workings of BERT (Bidirectional Encoder Representations from Transformers) [25], a system that leverages a multi-layer bidirectional transformer

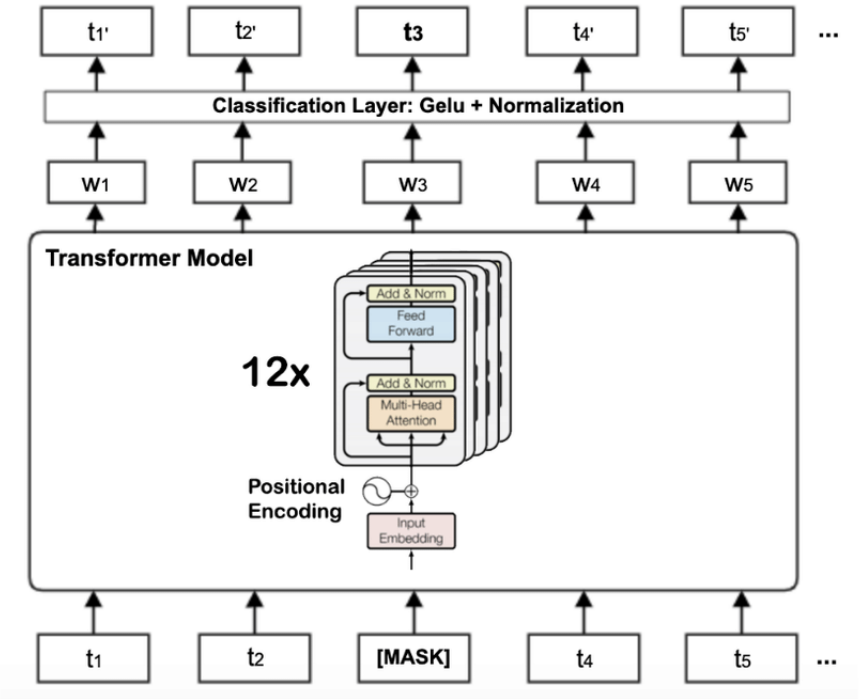

Figure 4.2: Architecture of the BERT model. [24]

encoder. BERT builds upon the foundation laid by the original implementation of the Transformer to tackle complex natural language processing tasks.

4.2.1 Background and Motivation

The development of the BERT model was motivated by the need to overcome limitations of existing language models at the time. In this section, we will explore why BERT was developed and what differentiates it from other models.
At the time of BERT's introduction, standard language models predominantly utilized unidirectional encoders. These encoders processed input text in a sequential manner, either from left to right or right to left. However, the authors of BERT identified several disadvantages with this unidirectional approach.
Unidirectional encoders, by processing text sequentially, only have access to the preceding context at each position. This limitation poses a challenge in capturing dependencies that require information from both past and future tokens. For tasks that involve sentence-level understanding, such as question answering, incorporating context from both directions becomes crucial.
To address this limitation, BERT adopts a bidirectional approach. By leveraging a bidirectional transformer encoder, BERT processes input text in both directions simultaneously. This enables each token to access both preceding and succeeding context, leading to a better understanding of the context and capturing a wider range of dependencies. This makes it well-suited for various natural language processing tasks. [25]

4.2.2 Architecture and Inner Workings

We will now delve into a detailed exploration of the architecture of the BERT model. Before we go into the neural network aspect, it is important to understand the initial tokenization step.

Tokenization

The input to BERT consists of a string that can encompass multiple sentences. This string is initially divided into its constituent words. These words undergo embedding using the WordPiece model [26], where each word is mapped to a predefined vector representation.
In cases where the input contains words that are not present in the vocabulary, which consists of 30,000 tokens, these words are further divided into subwords that do exist in the vocabulary. For instance, if the word "playing" is absent from the vocabulary, it would be split into two words: "play" and "##ing". The double pound symbol (##) signifies that the subword is part of a larger word.

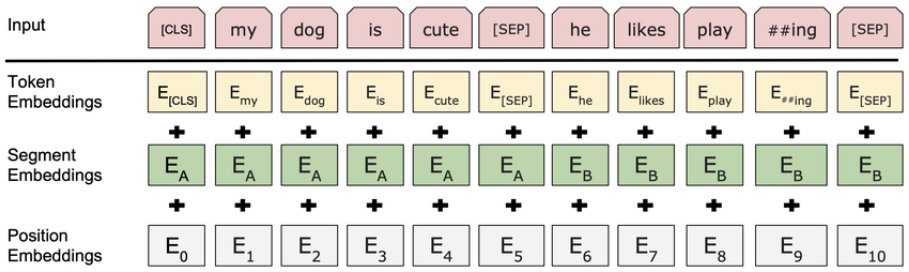

Figure 4.3: BERT input representation. [25]

Within BERT's architecture, there are special tokens that play distinctive roles. Examples are the *[CLS]* token, which indicates the start of a sequence, and the *[SEP]* token, which denotes the separation between two sentences within the input.
After the string has been split up into tokens, which are limited to a maximum of 512 tokens in BERT, all tokens extracted from the input sentence are mapped to vector representations known as token embeddings. This token embedding captures the semantic meaning of each token within the sentence. Additionally, a segment embedding is added to the token embeddings. Together with the [SEP] token, these segment embeddings assist the model in discerning which sentence each word belongs to.
To incorporate positional information, a position embedding is added to each token embedding, indicating the token's position within the sequence. This positional encoding enables the model to consider the sequential arrangement of the tokens, similar to what we have observed in the Transformer model.
Upon completing these steps, the input embeddings are obtained, which are constructed by summing the token embeddings, segment embeddings, and position embeddings. These embeddings serve as the foundation for subsequent processing and comprehension by the BERT model. It is important to clarify that these embeddings themselves do not inherently contain contextual information yet. The task of incorporating context lies with the self-attention mechanism within the neural network component of the model, which we are going to look at next. [25]

Neural Network Architecture

BERT, as mentioned earlier, is a multi-layer bidirectional transformer encoder that builds upon the original transformer implementation, sharing many similarities. Originally two versions of BERT were released: $\mathbf{BERT_{BASE}}$ and $\mathbf{BERT_{LARGE}}$. From here on we will only focus on the smaller $\mathbf{BERT_{BASE}}$ model, as in the classifier implementation we also use a smaller version of the model.
BERT is composed of 12 transformer encoders, stacked on top of each other, whereas the original transformer model had only 6. This increased depth is illustrated in Figure 4.3. Within each encoder, there are 12 attention heads, surpassing the 8 used in the transformer model. Following the attention heads, there is a feed-forward layer, which in BERT has double the hidden size compared to the original transformer.
Each encoder in BERT processes input and produces output vectors with a combined size of 768 elements. Consequently, BERT operates with a hidden size of 768 units. In total, BERT consists of a total of 110 million parameters. [25]

4.2.3 Pre-training

In this section, we look at the pre-training phase of BERT where the model learns to comprehend language. This involves exploring the tasks it is trained on and the data utilized in the training process. Specifically, BERT undergoes training on two key tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In this section, we will begin by delving into the MLM task, which focuses on predicting masked words within a given context.

Masked Language Modeling

In the MLM task, approximately 15% of the input tokens are randomly replaced with a special mask token, *[MASK]*. This forces the network to learn the most suitable replacement for the masked token. To achieve this, a feed-forward neural network is appended to each output of the last encoder layer.
These additional feed-forward layers include an output layer encompassing all English words. The outputs of these layers are then passed through a softmax activation function, resulting in a probability distribution across all English words. The word with the highest probability becomes the substitute for the masked token. An illustration of the MLM task is given in Figure 4.4

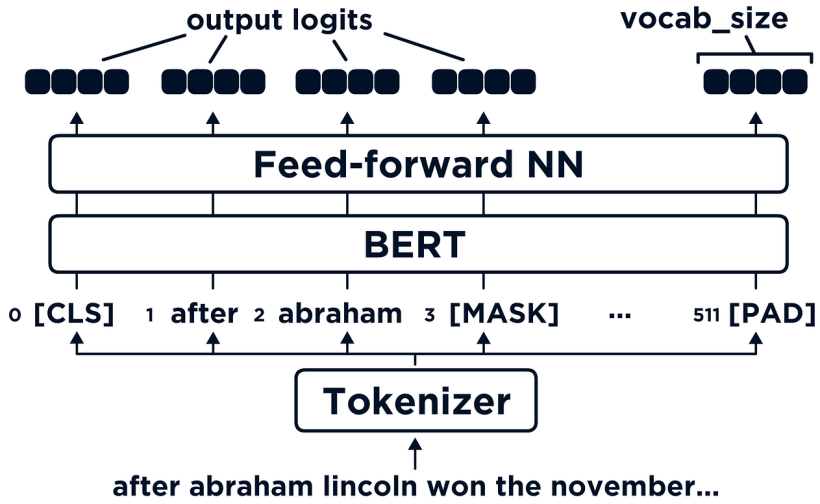

Figure 4.4: Masked Language Modeling in BERT. [27]

This process allows the model to be trained bidirectionally, leveraging the entire context of the sentence, including words both to the left and right of the masked token. The model develops contextual understanding, capturing the interdependencies between words.
It could however potentially learn to excel only in predicting masked tokens, which may hinder its performance when fine-tuned for different tasks lacking masked words. To address this, a small percentage of the chosen words for masking undergo various transformations:

- 80% of the words are masked with the *[MASK]* token.
- 10% of the words are substituted with random words.
- The remaining 10% of the words remain unchanged.

By incorporating these variations, the model is compelled to maintain robust embeddings for all tokens. This helps mitigate potential performance degradation when fine-tuning BERT for different downstream tasks. [25]

Next Sentence Prediction

During the pre-training phase, the model undergoes another procedure that enables it to understand the relationship between sentences. This procedure is formulated as a binary classification problem. Randomly selected sentences (referred to as A and B) are chosen from the corpus. In half of the instances, sentence B genuinely follows sentence A, while in the remaining cases, this sequential relationship does not hold. The model utilizes the output of the *[CLS]* token as the binary classification output for this task. [25]

Pretraining Data

The data used for pretraining the model consists of two major sources: the BooksCorpus and the English Wikipedia. These large text collections provide a rich and diverse range of linguistic content for the model to learn from. The BooksCorpus dataset encompasses a substantial 800 million words, while the English Wikipedia dataset offers an even larger volume of 2,500 million words.
By training on such large and varied textual data, the model gains a comprehensive understanding of language. [25]

4.2.4 Fine-tuning

The pretrained BERT model offers the flexibility to be fine-tuned for a wide range of tasks. To accomplish this, the outputs of the pretrained model are passed through a simpler network that specializes in finetuning the model for a specific task. By adjusting the weights of this new network during the fine-tuning process, BERT becomes specifically tuned for the desired task. Compared to the pre-training procedure, where the entire model is trained from scratch, this fine-tuning process is significantly simpler and faster.
This simple approach of adjusting the weights of the added task-specific network makes BERT an attractive choice for tackling various NLP problems. It makes it possible to adapt the powerful pretrained BERT model to specific tasks, without the need for extensive training time and computational resources. [25]

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Figure 4.5: GLUE Test results for BERT. [25]

4.2.5 Evaluation and Performance

The performance of BERT was evaluated through experiments conducted on the GLUE benchmark, which encompasses a diverse set of NLP tasks.
As seen in Figure 4.5 BERT demonstrated impressive results across all evaluated tasks. Its performance was compared to prior state-of-the-art systems such as OpenAI GPT and ELMo. Notably, BERT consistently outperformed these systems, achieving superior accuracy and outperforming them by a significant margin.
It is worth mentioning that $\mathbf{BERT_{LARGE}}$ consistently exhibited better performance than $\mathbf{BERT_{BASE}}$, particularly when dealing with limited training data.
The results obtained from these experiments emphasize the superior performance of BERT across a wide range of NLP tasks. [25]

4.3 RoBERTa: Robustly Optimized BERT Pretraining Approach

In 2019, a team at Facebook AI Research[1] conducted a replication study of the BERT model. During this study, they developed an enhanced version of BERT known as RoBERTa (Robustly Optimized BERT Pretraining Approach) [28]. RoBERTa surpasses the performance of the original BERT model and achieves state-of-the-art results in various natural language processing tasks. In the following sections, we will explore the different design choices that were implemented in RoBERTa, comparing them to the original BERT model.

4.3.1 Pre-training

The primary distinction between RoBERTa and BERT lies in their pre-training approaches. Although the network architecture remains unchanged, the RoBERTa team conducted experiments with various variations of the MLM and NSP procedures and also tokenization. These variations were aimed at further improving the model's language understanding capabilities. [28]

Dynamic Masked Language Modeling

In the original BERT model, masked language modeling (MLM) is performed in a static manner. This means that the masking of tokens is done once during the preprocessing stage. However, a limitation of this approach is that during training, the network would encounter the same masked training sequences multiple times as they are fed into it unchanged at each epoch.
To address this issue, the original BERT model duplicates each training instance ten times and applies a different mask to each duplicate. This technique ensures that the model sees variations of the masked training sequence. However, even with this approach, each training sequence is still observed four times with the same mask over the course of 40 epochs.
To overcome this limitation, RoBERTa introduced dynamic masking. With dynamic masking, a new masking pattern is generated every time a sequence is fed into the model. This approach is crucial when training the model for a larger number of epochs or with a larger dataset. The dynamic masking approach used in RoBERTa is comparable or slightly better than the static masking approach employed in the original BERT model. [28]

Necessity of Next Sentence Prediction

Next Sentence Prediction (NSP) has played a significant role in the original BERT model's training process. It was initially considered crucial for achieving high performance. The RoBERTa team however, arrived at a different conclusion.
Four different training formats were experimented with, and the findings suggest that the best-performing formats are the ones where NSP is not performed. Instead, sentences are sampled contiguously from a document until the maximum input size of 512 tokens is reached and then MLM is performed without the NSP task. [28]

[1] `https://ai.meta.com/research/`

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-

Figure 4.6: GLUE Test results for RoBERTa. [28]

Effect of Batch Size on Model Performance

The study also investigated the impact of different batch sizes and learning rates on the performance of the model. This investigation was motivated by previous research [29], which demonstrated the potential benefits of using larger batch sizes for Neural Machine Translation.
The original BERT model was trained with a batch size of 256 and 1 million steps. The researchers wanted to explore the effects of different batch sizes while keeping the computational cost the same. By measuring the model's performance under various batch size configurations and adjusting the number of steps accordingly, they aimed to identify the optimal setup.
After conducting experiments on three different configurations, the study concluded that a batch size of 2k, a learning rate of 7e-4, and 125k steps produced the best outcomes. This configuration not only improved the model's performance on the given tasks but increased parallelizability, allowing for more efficient training. [28]

Expansion of Pretraining Data

In addition to modifying the pretraining tasks and hyperparameters, the training data was expanded significantly.
In addition to the BookCorpus and English Wikipedia corpora, which together amounted to 16GB of text, several other datasets were included. These datasets included the *CC-NEWS* dataset, comprising 63 million English news articles with a total size of 76GB, the *OPENWEBTEXT* dataset consisting of content from various websites, totaling 36GB, and the *STORIES* dataset, totaling 31GB. The incorporation of these additional datasets resulted in a combined training corpus size of 160GB. [28]

4.3.2 Evaluation and Performance

The performance of the RoBERTa model was assessed using various metrics and comparisons with other state-of-the-art models for different natural language processing tasks seen in Figure 4.6.
It was determined that the best performing configuration of RoBERTa utilized a training data corpus of 160GB, a batch size of 8k, and a total of 500K training steps. This configuration resulted in superior performance compared to other models. It significantly outpermorms the original $\mathbf{BERT_{LARGE}}$ model and also outperforms the $\mathbf{XLNet_{LARGE}}$ model across a range of NLP tasks.
An interesting observation made by the authors was that even the longest-trained RoBERTa model did not exhibit overfitting. This finding suggests that the model could potentially benefit from additional training, indicating the potential for further performance improvements.
The superior performance of RoBERTa in various language tasks, proves its effectiveness as a powerful language model. and the evaluation results confirmed the efficacy of the modifications and strategies employed in RoBERTa's training, leading to improved language understanding and representation capabilities. [28]

4.4 DistilRoBERTa

Due to the convincing results RoBERTa achieved, it was decided to use it as the basis for our classifier. RoBERTa however is, as we've just seen, a very big model with lots of parameters, which makes it somewhat slow compared to smaller models. This poses a problem for our use case, as our ultimate objective is to develop an interactive website that can deliver results to users quickly and efficiently. Therefore, it is crucial for our classifier to perform inference rapidly.
To achieve this goal, it was decided to utilize the DistilRoBERTa model[2] by Hugging Face. DistilRoBERTa is a compressed version of the original RoBERTa model that maintains its high performance while significantly reducing its size. This compressed model allows us to achieve a balance between resource efficiency and accuracy. Before delving into the details of our fine-tuning architecture, let us very briefly explore the concept of knowledge distillation which is the key to the success of DistilRoBERTa. [30]

[2] `https://huggingface.co/distilroberta-base`

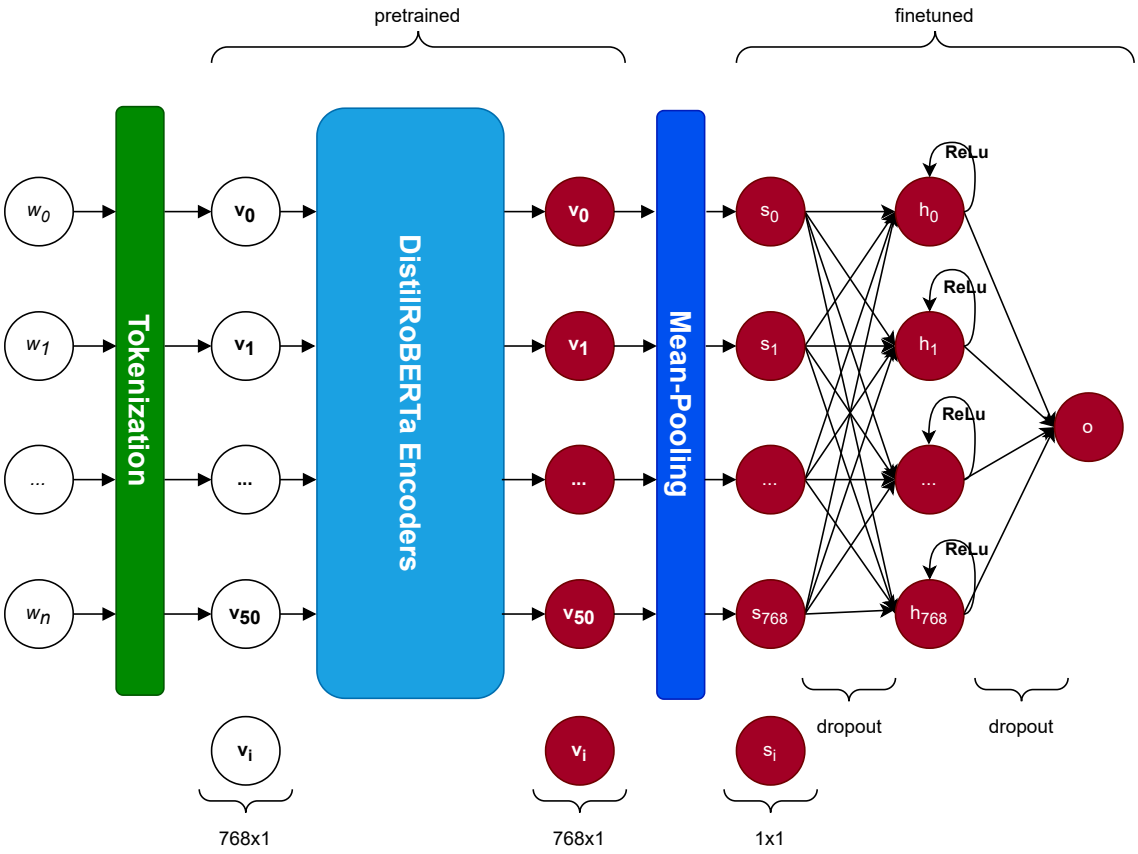

Figure 4.7: Architecture of the classifier.

4.4.1 Knowledge Distillation

Knowledge distillation is a compression technique designed to enable a smaller model, referred to as the student, to replicate the performance of a larger model, known as the teacher. This process involves incorporating the teacher's predicted output distribution into the training of the student model. This is achieved through the use of a distillation loss in addition to the standard supervised training loss.
The intuition behind employing this distillation loss lies in the fact that the teacher's output distribution provides more information to the student than just the one-hot-encoded training distribution alone. By leveraging the teacher's knowledge and understanding, the student model can learn more efficiently and capture the nuances present in the teacher's predictions. [31] [32]

4.4.2 DistilRoBERTa

DistilRoBERTa, as indicated by its name, incorporates the knowledge distillation process to distill the knowledge from RoBERTa. In addition, it introduces some modifications to the RoBERTa model. These modifications include the removal of certain tokenization steps, a reduction in the number of layers, and the utilization of optimized linear algebra frameworks for linear layers and layer normalization.
These modifications have been very successful, as demonstrated by the following results. DistilRoBERTa achieves an impressive 95% of the performance of RoBERTa-base on the GLUE benchmark. DistilRoBERTa also has significant advantages in terms of efficiency. It achieves twice the inference speed while being 35% smaller in size compared to RoBERTa-base.
This exceptional balance between performance and efficiency makes DistilRoBERTa a desirable choice for our use case. [30]

4.5 Finetuned DistilRoBERTa For Binary Future-related Sentence Classification

In figure 4.7, we can observe the high-level architecture of our classifier. In the subsequent sections, we will analyze the architecture's individual components, providing a detailed explanation and rationale behind the design choices.

4.5.1 Optimized Tokenization for Reduced Complexity

Firstly, from the figure it is evident that the number of tokens, also known as the sequence length, has been reduced to 50 tokens, thereby lowering the computational complexity and further increasing performance of the model. Considering that the model only processes one sentence at a time, and given that the average sentence length in our dataset is 20 words, this is sufficient.

4.5.2 Mean-Pooling the DistilRoBERTa Outputs

After the tokenization step, we obtain 50 embedding vectors, each having a dimension of 768. These vectors are then passed through the encoders of DistilRoBERTa. As a result, we obtain the same number of vectors with the same dimension, but now these vectors contain contextual information from the sentence. This contextual information, computed with attention, is visually represented by the red color in the figure.
While we now have contextualized embedding vectors for each input token, our goal is to classify the entire sentence as a whole. Therefore, it would be beneficial to have a single embedding vector that represents the entire sentence. This would also reduce the size of our finetuning feed-forward network. To achieve this, we compute the mean over all 50 embedding vectors. There are other ways to do this, such as utiliting the *[CLS]* token, however HuggingFace recommends employing mean pooling[3].
After the mean pooling operation, we obtain a single embedding vector of dimensionality 768. This vector captures the overall meaning and essence of the entire sentence, enabling us to perform efficient sentence-level classification on it next.

4.5.3 Feed Forward Fine-Tuning Network

The output of the mean-pooling operation serves as the input layer for our finetuning feed forward network. This network comprises a single hidden layer and a single output node, as we do binary classification. To introduce non-linearity, the hidden layer nodes are passed through a ReLU activation function. Additionally, dropout layers are incorporated before and after the hidden layer to mitigate overfitting and enhance generalization.

Training Details

During the training phase, the weights of the finetuning network were initialized randomly. It is important to note that the weights of the pretrained model were frozen completely during the fine-tuning process, so they are not part of the training process. We employ the binary cross entropy with logits loss as our loss function, which implicitly applies a sigmoid activation to the model output. The model was trained on our dataset over the course of 14 epochs with the following configurations:

- Batch size: 8
- Learning rate: $1.5e^{-6}$
- Warmup: 0.2
- Weight decay: 0.001

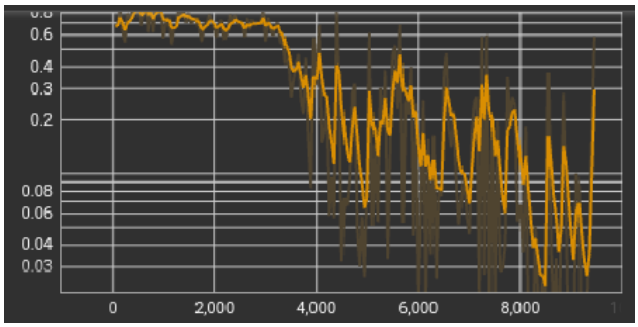

Figure 4.8: Training Loss of the Final Model.

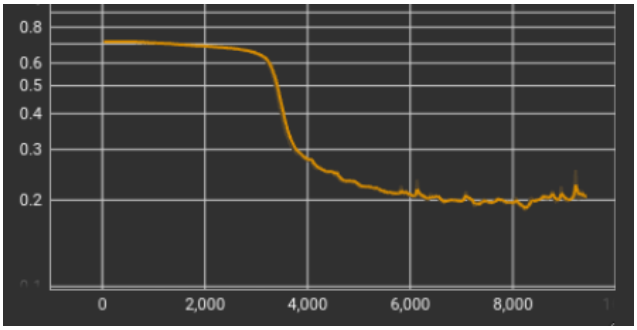

Figure 4.9: Validation Loss of the Final Model.

[3] https://huggingface.co/transformers/v4.7.0/model-doc/roberta.html

Validation Details

We employed 10-fold cross-validation with early stopping to obtain performance estimates for the model. The training and validation losses for each fold are illustrated in Figure 4.10 and Figure 4.11, respectively. The following metrics show the average scores we achieved:

- Accuracy: 0.965
- Precision: 0.953
- Recall: 0.98
- F1-score: 0.97
- AUC-ROC: 0.964

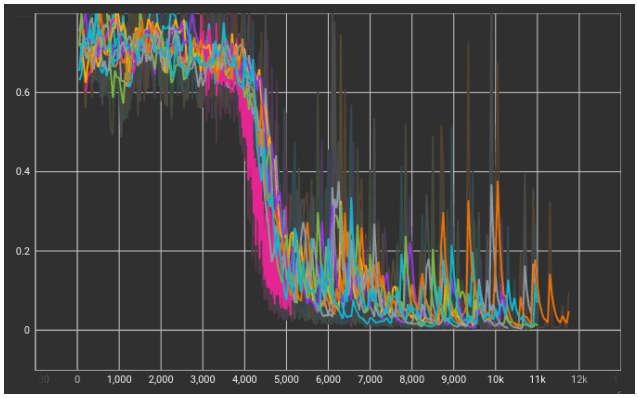

Figure 4.10: Training Losses of the 10 fold cross-validation process.

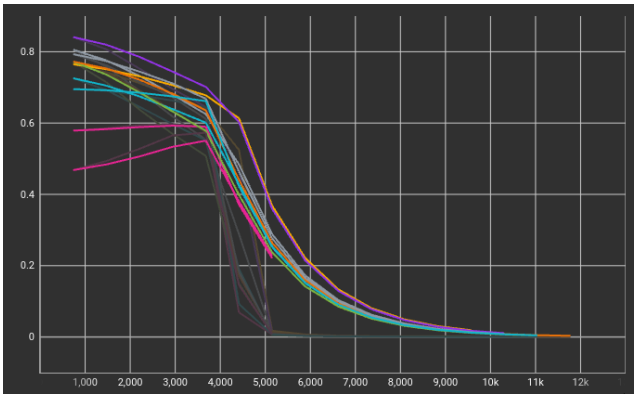

Figure 4.11: Validation Losses of the 10 fold cross-validation process.

Inference

The prediction process involves the classification of sentences extracted from news articles into either future-related or non-future-related sentences. First, all sentences extracted from news articles are subjected to classification using the trained model. The model assigns a confidence score between 0 and 1 to each sentence, indicating the likelihood of it containing future-related content. These confidence scores are then stored and analyzed.
Next, only sentences with a confidence score equal to or above 0.9 are considered as future-related sentences. They are then forwarded to the topic modeling component for further processing.
For sentences with confidence scores below 0.9, they are classified as non-future sentences. These sentences are stored in a separate database table. They can be later reviewed and manually inspected. Depending on the analysis, they may be added to the training dataset for further improvement of the model. Also the future-related sentences are stored in the database and may be later added to the training dataset.

Chapter 5

Topic Modeling

After downloading articles and classifying their sentences as future-related or not, the future-related sentences should be shown to the user. However, simply presenting these sentences individually may not provide a comprehensive understanding of the data. To address this, we want to group related sentences together. Among various grouping strategies, organizing sentences by topic emerges as the most logical choice. By clustering sentences that share common content, we can present interconnected information to the user. So the primary objective of this chapter is to explore the topic modeling approach employed in our system.

5.1 Shortcomings of Conventional Models

In the field of topic modeling, conventional models have long relied on representing documents as a bag-of-words, employing techniques such as Latent Dirichlet Allocation (LDA) [33]. One of the primary limitations of those conventional models is their failure to capture the semantic connections that exist between words. By treating a document as a collection of independent words, these models disregard the contextual information that can be obtained by analyizing the interplay of words within a sentence.
The rise of text embedding models revolutionized the way we understand and analyze textual data. Leveraging the success of these text embedding models, researchers began incorporating them into the realm of topic modeling.
One notable approach that bridges the gap between text embedding models and topic modeling is Top2Vec [34]. Instead of relying solely on the traditional bag-of-words representation, Top2Vec adopts a more comprehensive strategy. It clusters document embeddings, which capture the essence of the entire document in a vector, and extracts topic representations by identifying words whose embeddings closely align with the centroid of each cluster. These words represent the salient topic of the respective cluster.
However, it is important to acknowledge that even approaches like Top2Vec have their limitations. One key problem arises from the assumption that words near the cluster centers are the most representative of the topic. While this assumption holds true in many cases, clusters in real-world data often deviate from a spherical shape around their centers.
The misalignment between the assumption of spherical clusters and the actual distribution of data points can result in misleading topic representations as the words in close proximity to the cluster centroid may not accurately capture the full range of concepts and themes encompassed by the cluster. This limitation highlights the need for more advanced techniques that can accommodate complex cluster shapes and provide more accurate representations of topics within the data. [35]

5.2 BERTopic

To address the limitations we have discussed, an innovative topic modeling approach called BERTopic [35] has emerged. It leverages clustering techniques and a class-based TF-IDF to overcome the challenges posed by traditional models.
BERTopic operates through a three-step process. In the following sections, we will explore each step to gain an understanding of BERTopic's functionality.

5.2.1 Document Embeddings

The initial step involves mapping documents to a vector space, which can be accomplished through an embedding process. In the case of BERTopic, a pre-trained language model called SBERT [36] is utilized for this purpose. As the name suggests, SBERT is based on BERT.

There is a significant distinction between BERT and SBERT in their respective embedding approaches. BERT focuses on generating word embeddings that capture the meaning and context of individual words within a sentence. It achieves this by considering the surrounding words and their relationships. [25] On the other hand, SBERT aims to map similar sentences to similar vectors in the embedding space, specifically using cosine similarity as a measure. [36]
While BERT can also generate sentence embeddings by employing pooling techniques or the [CLS] token, studies [36] have shown that this approach produces subpar results. In contrast, SBERT has achieved state-of-the-art performance on various sentence embedding tasks, making it the default choice for BERTopic.
However, it is important to note that it is possible to replace SBERT with alternative models if desired. BERTopic allows flexibility in selecting different models for generating sentence embeddings. [35]

5.2.2 Document Clustering

In the task of document clustering, a challenge arises as the dimensionality of the data increases. At higher dimensions, the distance to the nearest point tends to approach the distance to the farthest point. Consequently, the concept of spatial locality becomes ambiguous, as the differences in distances become negligible.
This issue becomes problematic when attempting to cluster the high dimensional sentence embeddings generated in the previous step. One effective approach is to reduce the dimensionality of the embeddings before performing the clustering process.
A notable technique employed by BERTopic, is the utilization of a data dimensionality reduction algorithm called UMAP [37].
After applying the dimensionality reduction, the reduced embeddings are ready for clustering. For this task, BERTopic employs the HDBSCAN algorithm [38]. HDBSCAN is a robust clustering algorithm that is capable of identifying clusters of varying densities in the data, making it suitable for the clustering of document embeddings.
By employing these sequential steps of dimensionality reduction and subsequent clustering, BERTopic offers a good solution for document clustering tasks. [35]

5.2.3 Topic Representation

Now that the data is organized into clusters, the next step is to extract topic representations from these clusters. To achieve this, a modified version of TF-IDF is used, which measures the importance of words in a document. Traditionally, TF-IDF considers the frequency of a term within a document and the number of documents that contain the term. However, in the modified approach, all documents belonging to the same cluster are combined into one document. This adjustment allows to capture the significance of words within clusters, rather than individual documents.
The inverse document frequency is then computed by taking the logarithm of the average number of words per cluster divided by the frequency of the term across all classes. To ensure a positive outcome, one is added to the computed result.
This modified approach makes it possible to model the importance of words within clusters. As a result, the most important words can be identified as representatives of each topic, leading to a better understanding of the topics. [35]

5.2.4 Configuration of BERTopic for Future-related Sentence Clustering

In this section, we see the concrete configuration of BERTopic in our project.
In order to achieve optimal results, certain parameters of BERTopic were changed from their default values. The specific configuration used in this project is outlined below:

- **Sentence Embedding Model:** To generate high-quality sentence embeddings, the "all-MiniLM-L6-v2" model from the SentenceTransformer library was chosen.
- **Vectorizer Model:** A CountVectorizer was utilized after the embedding process to remove stopwords from the text data.
- **Top Words per Topic:** BERTopic was configured to provide a concise summary of each topic by extracting the top 10 most representative words associated with each cluster.
- **Minimum Topic Size:** The minimum topic size was adjusted to five in order to address the issue of having excessively large topics containing multiple subtopics. By reducing the minimum topic size, the granularity of the topics was increased.
- **Dimensionality Reduction Model:** UMAP was used for dimensionality reduction.
- **Clustering Model:** HDBSCAN served as the clustering method.

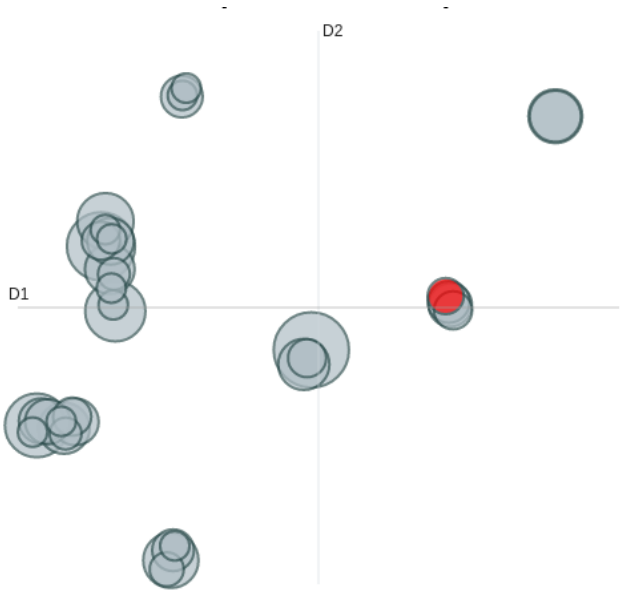

Figure 5.1: Intertopic Distance Map for Sentences from Ukraine-related News Articles using BERTopic.

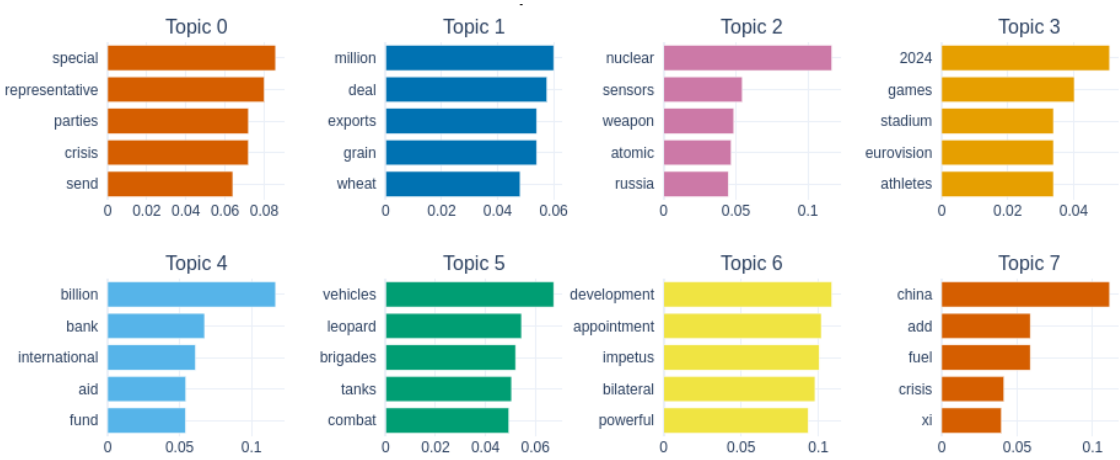

Figure 5.2: Topic Representation and c-TF-IDF Scores Generated by BERTopic.

Chapter 6

Temporal Expression Identification and Extraction

An important objective of this work is to associate sentences related to the future with a coherent timeline. To achieve this, it is necessary to analyze sentences for temporal expressions. Once this temporal information is identified, it is extracted and mapped to a specific date. Then it is possible to map a sentence to a concrete point in time and therefore to a concrete point on a timeline.

6.1 SUTime - Temporal Tagger

A system designed for this specific task is referred to as a 'temporal tagger'. In this work, the rule-based temporal tagger SUTime [39] is used. Despite its inception in 2012, SUTime continues to excel in its functionality.

6.1.1 Inner Workings

SUTime employs a systematic approach structured around three stages:

1. In the first stage, the system analyzes the the text to identify words that could potentially form numerical expressions.
2. Having identified numerical expressions, the system can use those in addition to the other words to start looking for simple temporal expressions.
3. In the final stage, basic temporal expressions are combined to create more complex expressions

This strategy is implemented through different types of patterns that SUTime supports for matching text: token patterns, string patterns, and time patterns.

- **Token patterns** operate over individual tokens in the text. A token is typically a word, number, punctuation or other meaningful unit of text. Token patterns in SUTime use a regular expression language (TOKEN-SREGEX) which is included in the Stanford CoreNLP package that works on tokens instead of raw text strings. This allows for the use of token annotations in the matching process. For example, a rule can be specified that matches a token like 'years' and maps it to a predefined duration type YEAR. More complex rules can then also be designed using macros or annotations that match specific types of tokens, such as numbers and duration units.
- **String patterns** operate at the string level, which means they consider the text as a whole and can match patterns anywhere in the text, regardless of token boundaries.
- **Time patterns**, on the other hand, are used for matching time-specific patterns over text that align with standard date/time formats.

Upon recognition of all temporal expressions, relative times are transformed into absolute references relative to the specified reference date. The results are returned in the form of TIMEX3[1] annotations, which is a standard to capture temporal information in natural language. [39]

[1] `http://timeml.org/site/publications/timeMLdocs/timeml_1.2.1.html`

6.1.2 Limitations

The capabilities of SUTime are not without their limitations. Although most of these constraints do not affect our specific use-cases, one stands out as particularly relevant.
While the tagger generally performs well, there are some problems with ambiguous references like for example *"On Tuesday, he revealed his decision, which will become official soon"*. The time-tagger maps *"on Tuesday"* to the coming Tuesday, which is in the future, even though the sentence is referring to a past event. Given the repeated occurrence of this issue during system testing, a decision was taken to exclude all temporal expressions referencing weekdays. [39]

6.1.3 Comparative Analysis of SUTime Against Other Temporal Tagging Systems

In order to evaluate the effectiveness of the SUTime temporal tagger, the authors conducted a series experiments comparing its performance against other state-of-the-art systems. The systems chosen for comparison were GUTime [40], HeidelTime [41], and TRIPS/TRIOS [42].
The authors discovered that both rule-based systems, represented by HeidelTime, and probabilistic systems, represented by TRIPS/TRIOS, exhibited comparable effectiveness in recognizing temporal expressions. This was reflected in their similar F1 scores, highlighting that rule-based approaches could capture a significant portion of temporal patterns.
However, among the systems under examination, SUTime emerged as the frontrunner in terms of performance, achieving the highest F1 score and recall for identifying temporal expressions. [39] This outcome underscores the ability of SUTime in accurately and comprehensively detecting temporal nuances within text.

Chapter 7

System Implementation and Architecture

In this chapter, we explore the detailed implementation of the frontend and backend components of the web application, which can be accessed with a standard browser at the following URL: `http://chronicle2050.regevson.com`. We explore the design choices made, the structure of the system, and the integration of various technologies and frameworks. This chapter highlights the development process, from the frontend user interface to the backend server architecture.

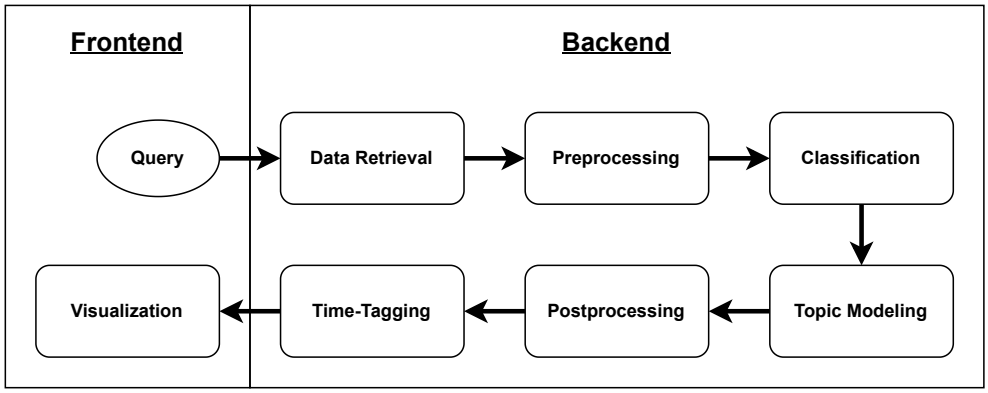

Figure 7.1: Schematic Overview of the System's Data Processing Pipeline.

7.1 Frontend Implementation

In this section, we explore the various aspects of the frontend implementation for our web application. We delve into the rationale behind the frontend framework selection, discuss the UI design and layout considerations, and examine specific features and functionalities implemented in the frontend.

7.1.1 Frontend Framework Selection

For this project, a frontend framework called Vue.js[1] (Vue) was chosen. Vue is a powerful framework that uses HTML, CSS, and JavaScript to enable efficient development of user interfaces. It accomplishes this by providing support for two fundamental features:

- **Declarative Rendering:** Vue extends standard HTML by introducing a template syntax that enables developers to embed JavaScript state directly into the HTML. This approach allows for more intuitive and concise UI development.
- **Reactivity:** One of Vues key strengths is its reactivity system. It monitors changes in the JavaScript state and automatically updates the Document Object Model (DOM) to reflect those changes. This feature eliminates the need for developers to manually manipulate the DOM, resulting in increased productivity and code maintainability.

[1] `https://vuejs.org/`

[43]
While there are numerous frameworks on the market offering similar functionality, Vue stands out as one of the most popular choices alongside Angular and React. All these modern frontend frameworks have large communities, active maintenance, extensive documentation, and good performance. However, Vue possesses specific strengths that were particularly important for this project, ultimately leading to its selection as the frontend framework.
First and foremost, Vue is highly praised for its simplicity and gentle learning curve. Its intuitive syntax and straightforward concepts make it easier for developers to grasp and start building applications quickly. This aspect was important for this project, where the ability to swiftly get started and incorporate changes efficiently was crucial. On the other hand, Angular and React are often considered somewhat more heavyweight, difficult, and complex in nature. [44]
So by choosing Vue, a simple and efficient development process was possible, enabling the rapid creation of the desired user interface while maintaining a manageable codebase.

7.1.2 UI Design and Layout

In this section, we discus about the planning and design of the user interface (UI).

Design Workflow

Before implementing the HTML/CSS layout, the UI was initially conceptualized and designed using Photoshop[2]. This workflow allowed for a quick visualization of the ideas before investing time into their actual implementation, which can be a more time-consuming process.

Settings and Input

Upon entry, the website shows a settings panel that provides the user with some adjustable configurations. Here, decisions about whether updates should be automatic or manual can be made. There's also the possibility to specify the quantity of articles and their time frame. A text field prompts users to specify an entity for exploration.

Timeline Visualization

The timeline section is one of the key components of the website. It offers valuable insights into the relationships between topics and the referenced dates within the associated sentences.

Axes The y-axis of the timeline contains all topics that contain sentences with temporal expressions. As already mentioned those temporal expressions could be phrases like "next August", "this Summer", ... The topics on the timeline are organized in a manner that reflects their relevance. This arrangement places more relevant topics towards the top of the y-axis, while less relevant ones are positioned towards the bottom. The specific criteria used to determine relevance will be explored and explained in detail in section 7.2.5.
On the other hand, the x-axis displays a series of dates, corresponding to the temporal expressions within the sentences.

Datapoints To provide a concrete example: a data point on the timeline, with coordinates (x, y), represents a collection of sentences belonging to topic y. These sentences contain temporal expressions referencing the date x.
Hovering over or clicking on a data point on the timeline reveals a tooltip section containing all sentences associated with the topic and the referenced date. Each sentence is accompanied by a numerical count, indicating how often it is mentioned in the news corpus. To avoid duplicate content, similar sentences offering the same information are intelligently removed to avoid redundancy and also count as mentions. A brief notification informs users about the omitted sentences due to similarity.
Furthermore, the section offers an option to display sentences linked to the topic, even if they lack temporal expressions and do not appear on the timeline. This provides a more comprehensive view of the topic and its context. To get more information, clicking on a sentence opens a new tab displaying the entire article from which the sentence was extracted. This feature enables users to access the full context, gaining deeper insights and making well-informed interpretations.

[2] `https://www.adobe.com/products/photoshop.html`

Word Cloud Visualization

The Word Cloud showcases sentences that lack any temporal expressions and is positioned beneath the timeline. Each distinct topic within the corpus receives its own dedicated cloud, displaying the most representative words associated with that topic. The size of each cloud dynamically adjusts based on the relevance of the corresponding topic, offering a visual cue about its significance.
Furthermore the Word Cloud is interactive. When users hover or click on a specific cloud, a tooltip emerges, revealing all the relevant sentences that belong to that particular topic. Just like with the timeline-tooltip, users can click on any sentence within the tooltip to instantly access its source article. Moreover, the number of times each sentence appears in the corpus is indicated next to it, providing additional context about its frequency and importance.

Charting Library

The timeline was created using ApexCharts[3], a charting library that enables the creation of interactive visualizations for web pages. To seamlessly integrate ApexCharts with Vue.js, a library called vue3-apexcharts[4] was utilized, which acts as a convenient wrapper library. This integration helps Vue developers by allowing them to interact with ApexCharts using the familiar Vue syntax.

7.1.3 Interaction with the Backend

The frontend and backend work together seamlessly to provide a good experience to the user. When the user clicks the "Explore" button, Axios[5], a popular JavaScript library for handling HTTP requests, comes into play. It triggers an HTTP GET request towards the backend's 'start/' endpoint, prompting the backend to begin the retrieval, classification, clustering, and tagging processes.
As the backend processes the data, the frontend polls the 'status'-endpoint, continuously receiving an array containing a checksum value and processing status information. This checksum value acts as a unique identifier for the current status of the database. Whenever a change occurs in the database, such as the availability of new sentences ready for presentation, the checksum value changes, and the frontend visualizes the new data on the timeline and word cloud. Furthermore, the statistics are updated to reflect the most current information available.

7.2 Backend Implementation

With a clear grasp of the layout and insight into the frontend-backend interaction, the focus now shifts to the backend implementation.

7.2.1 Django Framework

The backend infrastructure was developed with the Django Framework[6]. It is an open-source platform and is built upon the foundation of the Python programming language, leveraging Python's principles and capabilities. Django empowers developers to swiftly translate their concepts into reality. One of its standout attributes lies in its scalability which allows developers to build applications that can handle increased user loads and growing datasets. While such features are present in many well-regarded frameworks, Django was chosen specifically due to its use of the Python programming language. As the classifier is implemented with PyTorch[7] in Python, it was logical to use a python-based framework like Django for the backend. Moreover, Django seamlessly integrates with a REST framework[8], making it very easy to create and expose endpoints. The framework's simplicity further contributed to its selection, making it an ideal choice to kickstart and quickly implement various functionalities.

7.2.2 Data Retrieval

The user initiates the process by specifying the desired number of articles for download, accompanied by a designated timeframe that dictates the acceptable publication dates. Then an HTTP request, containing the

[3] https://apexcharts.com/
[4] https://github.com/apexcharts/vue3-apexcharts
[5] https://axios-http.com/
[6] https://www.djangoproject.com/
[7] https://pytorch.org/
[8] https://www.django-rest-framework.org/

user's specified configuration, is dispatched to the server, which in turn forwards this to the News API[9] which downloads the articles.
Following the data retrieval, irrelevant attributes are eliminated, leaving behind only the core elements of each retrieved article. The articles are stored in a new database table. This table stores details, such as article titles, content, publication timestamps, associated topics, and hyperlinks of the articles.

7.2.3 Preprocessing

In the preprocessing stage articles with an empty article field are removed from the dataset. Next, the NLTK library[10] is employed to tokenize the remaining articles into sentences.
Following tokenization, the length of each sentence is calculated, and any sentences with a length below 30 are filtered out. This step helps focus on sentences that provide substantial content and minimize noise in the dataset.
To address duplicate sentences, a deduplication process is applied. Duplicate sentences are removed, while keeping track of the count of duplicates for each sentence. This count becomes valuable in later stages for computing the relevance of individual sentences.
Finally, the processed sentences are stored back into the database, ensuring a clean and refined dataset ready for subsequent processing.

7.2.4 Classification

After the articles are split up into sentences and subjected to preprocessing, the classification step itself is executed. This process takes place remotely using Google Colab[11], due to its offering of high-speed GPUs like the V100 and A100. This is important because of the web application's online inference requirement, demanding optimal speed. Yet, it's important to acknowledge a limitation - the web application is only functional when the Colab notebook is running. Thus the accessibility of the service is restricted and can therefore only be used for demonstration and development purposes. However, as time progresses, there is a potential to replace the Colab server with a web server that has a GPU and offers better availability. The implementation of the DistilRoBERTa model, augmented with a finetuning network, was done using PyTorch Lightning[12]. This framework simplified the implementation by minimizing code and simplifying tasks like data loading and model training. After classification, sentences with a model confidence level over 0.8 are stored within a designated database table, ready for the subsequent clustering phase.

7.2.5 Topic Modeling and Postprocessing

The clustering step also takes place within the Google Colab environment, leveraging its powerful GPU capabilities for better speeds.
The clustering step is straightforward due to the simple nature of BERTopic. As detailed in Section 5.2.4, the specific parameter values used have already been established. BERTopic generates an array for each topic, filled with sentences belonging to that topic, alongside a corresponding array containing the sentence embeddings for further analysis. The representative topic keywords can be easily obtained by simply calling the corresponding function.
To improve the quality of topic modeling results and rank sentences based on their relevance, we postprocess the results in two ways. First, we remove outliers from topics by matching the topic keywords against the words in each sentence. If a sentence contains fewer than three topic keywords, it is removed from the dataset. The next step eliminates redundancy within a topic and evaluates the relevance of each sentence. Using the previously generated sentence embeddings from the topic modeling step, we calculate the cosine similarity between all sentence pairs within the same topic. When two sentences have a cosine similarity above 0.8, they are considered duplicates. The shorter sentence is discarded, and the duplicate count of the longer sentence is incremented. This duplicate count serves as a measure of sentence relevance. The greater the number of duplicates, the more frequently the sentence appeared in news articles and the more relevant it might be.
The results of this process find their place in a dedicated database table. This table stores the processed sentences alongside their respective topics and duplicate counts. These sentences await the subsequent time-tagging process.

[9] `https://www.newscatcherapi.com/`
[10] `https://www.nltk.org/`
[11] `https://colab.google/`
[12] `https://lightning.ai/docs/pytorch/stable/`

7.2.6 Time-Tagging

The final time-tagging step is structured as follows. First the sentences are retrieved from the database. These sentences are then passed to the SUTime parse function alongside their associated publishing date, yielding a JSON object as output. This object either contains the identified temporal expression along with its type or it is empty in the case of no such expression being present in the sentence.
Due to the limitations of SUTime laid out in 6.1.2, only temporal expressions of the *"DATE"* type are considered. Despite its name, this type encompasses not only specific dates but also expressions corresponding to periods within the year like *"in three months," "next year,"* and *"in the next decade"*.
SUTime implicitly converts temporal expressions into dates. However, in instances where direct conversion is not possible, such as in *"next summer"* or *"in the next decade"*, manual intervention is needed. For instance, *"next summer"* is manually translated to *"07-15"*, while *"in the next decade"* gets mapped to the current date with five years added to it.
Expressions referencing the past are omitted. In cases where sentences contain multiple temporal expressions, the one aligning with the furthest future date is selected.
Upon successful identification and mapping, the extracted dates are added to the database (containing the clustered sentences) as a new column. This marks the end of the backend's data processing workflow.

Chapter 8

Evaluation

To assess the effectiveness of our system, we developed a six-task exercise and presented it to six users of varying age and technical affinity. Each user had to complete the exercise with one of three entities. These tasks were designed to examine the timeline and word cloud components. The tasks included:

- 1) Summarizing the entity's future outlook.
- 2) Determining which topic and date have the highest/lowest relevance and analyzing the overall ranking.
- 3) Engaging with the timeline to find incorrectly resolved temporal expressions.
- 4) Summarizing the content of the most relevant word cloud.
- 5) Examining the word clouds for duplicate content.
- 6) Examining both components for sentences without future-related content.

The users provided lengthy summaries, implying a wide range of predictions for upcoming years. The color-coding of datapoints managed to successfully convey relevance, but users noticed that nearer predictions were generally marked as more relevant, while key distant predictions often had lower relevance scores. Some users found this to be a problem, as distant events are often more significant and require more time to unfold. The time-tagging was generally accurate but struggled with some past predictions like *"In 2019, he correctly predicted that a pandemic would occur by the end of the year"*. This is because the tagger resolves temporal expressions relative to the parent article's publishing date, and does not understand that *"in 2019"* should be the anchor date in this sentence. Some users found similar sentences in the same cluster, but the information they provided was slightly different. This provided additional context, which was well-received.
Overall, the system received strong approval from its users. On a five-star scale, the following results were obtained:

- Topic quality: 4 star average
- Classification quality: 4 star average
- Time-tagging quality: 3 star average
- Ranking: 3.5 star average
- Visualization: 4.5 star average

Users were particularly satisfied with the ability of the system to provide information about an entity's future plans, events, and decisions. However, the biggest critique was the system's speed, especially the time it takes to retrieve the initial results.

Chapter 9

Conclusion

In this work we have developed an approach and implemented a working system for extracting and visualizing future events on timelines for user queries.
By developing a multi-source dataset of 6,800 labeled sentences, we fine-tuned a DistilRoBERTa model to identify future references. We then used BERTopic for topic modeling and SUTime for time-tagging to extract topics from the data and detect and resolve temporal expressions. A dedicated postprocessing step improved cluster quality and ranked sentences according to their relevance. We also designed an intuitive interface that presents the results with an interactive timeline and word clouds.
Apart from exploring the implementation details of our system, we also gained insights into the inner workings of the models that were utilized in our system. We followed the evolution of the Transformer to more advanced language models like RoBERTa and DistilRoBERTa. We explored the shortcomings of conventional topic modeling approaches and the advantages of BERTopic. We also examined the functionality of the SUTime temporal tagger and its limitations.
In the end a user study confirmed the system's ability to offer a detailed overview of the future, while also highlighting potential areas for improvement, especially speed.

Bibliography

[1] Ricardo Baeza-Yates. Searching the future. In *SIGIR Workshop MF/IR*, volume 5, 2005.

[2] Inderjeet Mani and George Wilson. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 69–76, Hong Kong, October 2000. Association for Computational Linguistics.

[3] Hideki Kawai, Adam Jatowt, Katsumi Tanaka, Kazuo Kunieda, and Keiji Yamada. ChronoSeeker: Search engine for future and past events. In *Proc. of the 4th ICUIMC*, ICUIMC '10. Association for Computing Machinery.

[4] Adam Jatowt, Hideki Kawai, Kensuke Kanazawa, Katsumi Tanaka, Kazuo Kunieda, and Keiji Yamada. Analyzing collective view of future, time-referenced events on the web. pages 1123–1124, 04 2010.

[5] Nattiya Kanhabua, Roi Blanco, and Michael Matthews. Ranking related news predictions. In *Proc. of the 34th ACM SIGIR*, SIGIR '11, pages 755–764. Association for Computing Machinery.

[6] Kensuke Kanazawa, Adam Jatowt, and Katsumi Tanaka. Improving retrieval of future-related information in text collections. In *IEEE/WIC/ACM 2011*, volume 1, pages 278–283.

[7] Aiming Ni, Jinho D. Choi, Jason Shepard, and P. Wolff. Computational exploration to linguistic structures of future: Classification and categorization. In *North American Chapter of the Association for Computational Linguistics*.

[8] Yoko Nakajima. Automatic extraction of references to future events from news articles using semantic and morphological information. In *Proc. of the 24th IJCAI*, IJCAI'15, pages 4385–4386. AAAI Press.

[9] Navya Yarrabelly and K. Karlapalem. Extracting predictive statements with their scope from news articles. In *ICWSM*.

[10] Yoko Nakajima, Michal Ptaszynski, Hirotoshi Honma, and Fumito Masui. A method for extraction of future reference sentences based on semantic role labeling. *IEICE Trans. Inf. Syst.*, 99-D(2):514–524, 2016.

[11] Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. Predicting the news of tomorrow using patterns in web search queries. In *IEEE/WIC/ACM 2008*, volume 1, pages 363–367.

[12] Kira Radinsky and Eric Horvitz. Mining the web to predict future events. In *Proc. of the 6th WSDM*, WSDM '13, pages 255–264. Association for Computing Machinery.

[13] Russell Swan and James Allan. Automatic generation of overview timelines. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, page 49–56, New York, NY, USA, 2000. Association for Computing Machinery.

[14] Adam Jatowt and Ching-man Au Yeung. Extracting collective expectations about the future from large text collections. In *Proc. of the 20th CIKM*, pages 1259–1264. ACM.

[15] James Allan, Rahul Gupta, and Vikas Khandelwal. Temporal summaries of new topics. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 10–18, New York, NY, USA, 2001. Association for Computing Machinery.

[16] Hai Leong Chieu and Yoong Keok Lee. Query based event extraction along a timeline. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, page 425–432, New York, NY, USA, 2004. Association for Computing Machinery.

[17] Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. Timeline generation through evolutionary trans-temporal summarization. In Regina Barzilay and Mark Johnson, editors, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 433–443, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[18] Julius Steen and Katja Markert. Abstractive timeline summarization. In *Proc. of the 2nd Workshop on New Frontiers in Summarization*, pages 21–31, Hong Kong, China, November 2019. Association for Computational Linguistics.

[19] Yi Yu, Adam Jatowt, Antoine Doucet, Kazunari Sugiyama, and Masatoshi Yoshikawa. Multi-TimeLine summarization (MTLS): Improving timeline summarization by generating multiple summaries. In *Proc. of the 59th ACL and 11th IJCNLP*, pages 377–387. Association for Computational Linguistics.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[21] OpenAI. Introducing chatgpt, 2023. Accessed: 2024-01-06.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[23] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers, 2021.

[24] Usama Khalid, Mirza Omer Beg, and Muhammad Umair Arshad. Rubert: A bilingual roman urdu bert using cross lingual transfer learning, 2021.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. cite arxiv:1810.04805Comment: 13 pages.

[26] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[27] James Briggs. Masked-language modeling with bert, 2021. Accessed: 2024-01-06.

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[29] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[30] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[31] Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery.

[32] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[33] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, mar 2003.

[34] Dimo Angelov. Top2vec: Distributed representations of topics. *CoRR*, abs/2008.09470, 2020.

[35] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure, 2022.

[36] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[37] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[38] Leland McInnes and John Healy. Accelerated hierarchical density based clustering. In *ICDMW2017*. IEEE, nov 2017.

[39] Angel X. Chang and Christopher Manning. SUTime: A library for recognizing and normalizing time expressions. In *Proc. of the 8th LREC'12)*, pages 3735–3740, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

[40] Inderjeet Mani. Recent developments in temporal information extraction. In Nicolas Nicolov, Kalina Bontcheva, Galia Angelova, and Ruslan Mitkov, editors, *Recent Advances in Natural Language Processing III, Selected Papers from RANLP 2003, Borovets, Bulgaria*, volume 260 of *Current Issues in Linguistic Theory (CILT)*, pages 45–60. John Benjamins, Amsterdam/Philadelphia, 2003.

[41] Jannik Strötgen and Michael Gertz. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In Katrin Erk and Carlo Strapparava, editors, *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, pages 321–324. The Association for Computer Linguistics, 2010.

[42] Naushad UzZaman and James F. Allen. Trips and trios system for tempeval-2: Extracting temporal information from text. In *International Workshop on Semantic Evaluation*, 2010.

[43] Evan You. The progressive javascript framework, 2024. Accessed: 2024-01-06.

[44] Katrin A. Kern. Comparing modern front-end frameworks. Bachelor's thesis, Johannes Kepler University Linz, Linz, Austria, March 2022. Bachelor's Program in Computer Science.

[45] Yusuke Yamamoto, Taro Tezuka, Adam Jatowt, and Katsumi Tanaka. Supporting judgment of fact trustworthiness considering temporal and sentimental aspects. In *Web Information Systems Engineering-WISE 2008: 9th International Conference, Auckland, New Zealand, September 1-3, 2008. Proceedings 9*, pages 206–220. Springer, 2008.